**Product category service:**

public class ProductCategoryService : BaseService<ProductCategory>, IBaseService<ProductCategory>

{

public ProductCategoryService(IServiceProvider provider)

: base(provider)

{

RuleFor(t => t.Title).Required().UniqAsync("The title of the product category must be unique.");

RuleFor(t => t.Code).UniqAsync("The code of the product category must be unique.")

.CustomValue(code =>

{

if (!code.HasValue())

return false;

return code.Length > 2;

}, "The product category code must be two digits at most.")

.Custom(pc =>

{

if (!pc.Code.HasValue())

return false;

return new ProductService(ServiceProvider).GetAll().Any(p => p.Code == pc.Code);

}, "A product with this code has been registered.");

}

public async Task IncOrderingAsync(ProductCategory productCategory)

{

var old = await SingleAsync(productCategory.Id);

var pre = await GetAll().Where(t => t.Ordering < old.Ordering).OrderByDescending(t => t.Ordering).FirstOrDefaultAsync();

if (pre != null)

{

var temp = old.Ordering;

old.Ordering = pre.Ordering;

pre.Ordering = temp;

await base.UpdateAsync(pre);

await base.UpdateAsync(old);

}

}

public async Task DecOrderingAsync(ProductCategory productCategory)

{

var old = await SingleAsync(productCategory.Id);

var next = await GetAll().Where(t => t.Ordering > old.Ordering).OrderBy(t => t.Ordering).FirstOrDefaultAsync();

if (next != null)

{

var temp = old.Ordering;

old.Ordering = next.Ordering;

next.Ordering = temp;

await base.UpdateAsync(next);

await base.UpdateAsync(old);

}

}

public async override Task<ProductCategory> AddAsync(ProductCategory entity)

{

entity.Ordering = (await GetAll().MaxAsync(t => (int?)t.Ordering)).GetValueOrDefault() + 1;

return await base.AddAsync(entity);

}

public async override Task UpdateAsync(ProductCategory entity)

{

var old = await SingleAsync(entity.Id);

entity.Ordering = old.Ordering;

await base.UpdateAsync(entity);

}

}